Une distribution Hadoop pour la visualisation de données de simulation massives

Benoit Lange
Projet Opale
INRIA Rhône-Alpes
38334 Saint-Ismier, France
benoit.lange@inria.fr

Toan Nguyen
Projet Opale
INRIA Rhône-Alpes
38334 Saint-Ismier, France
toan.nguyen@inria.fr

RESUME
Dans cet article, nous allons nous intéresser à présenter notre outil de déploiement de distribution Hadoop. Cet outil a été développé dans le cadre du projet VELaSSCo (Visualization for Extremely LArge-Scale Scientific Computing – EC FP7). Dans ce projet, nous cherchons à développer une plateforme de stockage de données spécifiquement adaptée pour les simulations mécaniques. Cette plateforme devra également respecter un certain nombre de besoins autour de la visualisation de ces données. Cette stratégie de stockage doit également être adaptée aux capacités matérielles des scientifiques, en effet la plupart de leur matériel est destiné à réaliser des calculs haute-performance et non à manipuler de grandes quantités d'informations.

Dans cet article, nous allons présenter en détails le projet VELaSSCo, puis nous nous attacherons à présenter notre outil de déploiement d'une distribution Hadoop sur n'importe quel type d'environnements. Enfin, nous évaluerons des exemples de distributions déployées par notre outil et myHadoop, en proposant plusieurs évaluations, basées sur des métriques adaptées à ce projet.

Mots Clés
Hadoop ; HPC ; centre de stockage; visualisation ; Hive ; HBase ; machines virtuelles ; conteneurs.

ACM Classification Keywords

C.2.4 Distributed Systems

INTRODUCTION
La communauté scientifique produit des données depuis de nombreuses années. Ces données sont principalement issues de deux sources : simulations et mesure réelles. Ces données produites représentent maintenant de très grandes quantités d'information et de nouveaux problèmes apparaissent. Il devient alors compliqué pour les scientifiques de comprendre et d'analyser efficacement ces données massives.

Cette évolution dans la consommation de la science s'est déroulée en plusieurs étapes qui sont présentées dans [1]. Les premiers travaux de recherche se sont intéressés à l'étude et la compréhension des phénomènes physiques (en utilisant une description). Puis, des théories ont commencé à être élaborées (la loi de Newton, équation de Maxwell, …). Et plus récemment, des modèles de calcul ont été mis en place pour valider ces théories. Ces modèles de calcul sont alors validés par des simulations informatiques. L'afflux massif d'informations issues de ces différentes étapes, a fait naître un nouveau paradigme autour de la science des données massives. Ainsi, grâce aux nouvelles capacités matérielles, il devient possible de valider chaque étape du modèle scientifique en produisant de plus en plus d'informations. Malgré tout cette production massive est limitée, notamment par les capacités de stockage et de calcul des systèmes informatiques. Dans le cadre du projet VELaSSCo, nous essayons de résoudre ce problème en proposant une solution de stockage spécifiquement destinée aux données scientifiques.

Le projet VELaSSCo (Visualization for Extremely LArge-Scale Scientific Computing – EC FP7) est un projet financé par l'Union Européenne. Il a pour but de stocker et analyser différents jeux de données produits par des moteurs de simulations. Ce type de logiciel est capable de produire de grandes quantités de données corrélées ; une simulation est composée de différents pas de temps qui représentent l'évolution des données de manière temporelle. Les données qui vont être traitées par cette plateforme sont composées de deux types : des résultats de simulation FEM (méthode à élément fini) et DEM (modèle à élément discret). Les algorithmes DEM sont des algorithmes basés sur des particules, qui calculent les interactions entre différents éléments dans le temps. Cette stratégie est notamment utilisée pour les simulations astrophysiques (N-core), la dynamique moléculaire, la simulation de fluide etc. Par exemple dans le cadre d'une simulation astrophysique, un pas de temps pour une simulation de 10 millions de particules peut être produit en 1.19 s avec une taille de 500 Mo, [2]. Ainsi, stocker plusieurs milliers de pas de temps devient alors rapidement contraignant. Pour les simulations FEM, les données stockées sont alors composées d'éléments de discrétisation d'un espace, généralement des tétraèdres. Ici aussi, la quantité d'informations produites est importante.

Pour notre infrastructure logicielle, nous avons cherché la meilleure stratégie logicielle possible afin de répondre à un certain nombre de nos besoins : extensibilité, passage à l'échelle, réduction des coûts de développement, etc. Ainsi, notre choix s'est tourné vers le Framework Hadoop qui possède un écosystème et des caractéristiques compatibles avec les besoins de notre projet. Notre choix s'est porté sur cette suite logicielle notamment pour sa forte capacité à être étendu : certains besoins de notre consortium nécessitent de supporter plusieurs types de stockage de l'information. Hadoop sera utilisé comme squelette de notre plateforme et les composants seront déployés dans cet environnement en fonction des besoins. Ces besoins seront contraints par les briques logicielles mais également par le matériel disponible. Ainsi, des stratégies de déploiement de distribution Hadoop ont vu le jour pour des clusters HPC, mais ces méthodes n'exploitent pas au mieux les capacités matérielles (par ex : myHadoop).

Dans cet article, nous proposons un outil de déploiement d'une distribution Hadoop spécialement configurée pour supporter les données de simulation FEM et DEM. Notre outil permet de déployer tous les composants nécessaires à une version optimisée d'Hadoop en fonction d'une description de l'architecture matérielle produite par l'utilisateur. Le but final est de proposer une plateforme spécialement destinée à gérer des données massives pour la visualisation.

Dans la section 2, nous présentons un état de l'art concernant la gestion de données massives. Dans la section 3, nous nous intéressons à présenter le projet VELaSSCo ainsi que notre outil de déploiement de distribution Hadoop. Enfin dans la section 4, nous présentons une comparaison détaillée des premiers résultats de notre plateforme (notamment pour la visualisation batch). Et enfin, nous présenterons nos conclusions ainsi que nos futurs travaux.

ETAT DE L'ART

Dans cette section, nous allons nous intéresser aux travaux précédents autour de la thématique BigData, mais également autour de l'application Hadoop.

Lorsque l'on parle de BigData, il est nécessaire d'évoquer les 3 dimensions de ce domaine. Ainsi, il est régit par la règle des 3Vs : Volume, Vitesse et Variété (cf. [3, 4]). Ce modèle peut-être étendu à 5Vs en ajoutant deux nouvelles dimensions : Valeur et Vérité. Mais le modèle traditionnel reste cloisonné aux trois dimensions principales. Le projet dans lequel nous sommes impliqués est fortement corrélé avec ces trois dimensions :

- Volume : dans le cadre des simulations actuelles, la quantité de données produites est équivalente à plusieurs Giga-octets, mais à l'horizon 2020, cette quantité d'information aura atteint le Peta-octets (10^{15} octets).
- Vitesse : l'architecture de calcul des ordinateurs a largement évolué, et ces nouveaux systèmes permettent maintenant de réaliser des calculs très rapidement. Les simulations bénéficient très largement de cet avantage en exploitant au mieux les nouveaux dispositifs (GPU, multithreading, etc.). La vitesse d'accès aux données est également un facteur important de notre projet, car nos outils de visualisation devront être temps réel.
- Variété : dans le cadre de ce projet, nous allons stocker des données issues de différentes sources, ainsi il est nécessaire de se diriger vers des solutions de stockage non structurées (données sous forme de tableau, modèles de bâtiments, métadonnées, etc.).

Les stratégies autour du BigData ont été très largement discutées dans différents domaines, mais le domaine de l'ingénierie ne s'y est que très peu intéressé, VELaSSCo a pour but de répondre à ce besoin. Un des acteurs principaux dans le domaine du BigData est la société Google. Ainsi, ils ont proposé, au cours de ces dernières années, plusieurs articles scientifiques de référence pour le domaine. Ils ont ainsi réintroduit des modèles de calcul existants en les adaptant à leur besoin de fouille de données pour les moteurs de recherche. Par exemple, un papier de Dean et Ghemawat présente le modèle d'extraction d'informations appelé Map/Reduce, [5]. La fonction *Map* cherche à extraire une information à partir d'un système de fichier distribué, tandis que la fonction *Reduce* va agréger les données des fonctions *Map*. Cette stratégie fortement corrélée à un système de fichiers distribué appelé *GFS* (Google File System) est présentée dans [7]. *GFS* est un système de fichiers distribué spécifiquement destiné à être exécuté sur des nœuds de calcul standard. Toutes les sécurités nécessaires : tolérance aux fautes, redondance, etc. sont gérées par la couche application. Pour ce faire, cette méthode s'appuie sur deux types de nœuds : masters (qui stockent la hiérarchie et la localisation des fichiers) et les nœuds esclaves qui stockent des fragments de fichiers. Certains besoins chez Google les ont poussé à développer un stockage orienté table (au lieu de fichiers) qui se nomment *BigTable*, [6]. Malheureusement, ces différents outils ne sont pas accessibles, et restent la propriété de Google.

Malgré tout, des solutions alternatives ont vu le jour. L'une des plus connues se nomme Hadoop, voir [8]. Il s'agit d'une des applications les plus utilisées dans le domaine du BigData. Cet outil fournit toutes les briques nécessaires pour répondre aux besoins émis par la communauté scientifique, mais également répond directement aux spécifications des outils de Google. Ainsi, cette distribution fournit tous les composants nécessaires pour déployer un écosystème distribué pour stocker des données et distribuer le calcul. Ce logiciel repose sur deux composants : un système de fichiers : *HDFS* (HaDoop File System) présenté en [9] et le modèle de calcul *MapReduce*, [10]. Hadoop a été

développé en Java dans l'optique d'être le plus extensible possible. Ainsi, est venu se greffer un certain nombre d'applications supplémentaires compatibles avec l'écosystème Hadoop. En comparaison avec Bigtable de Google, *HBase* a été développé, [11, 12]. Des nombreux autres outils ont également été proposés afin de répondre à des besoins spécifiques : de stockage, de traitement, etc.

D'autres écosystèmes similaires à Hadoop ont été développés. Certains sont spécialisés dans une implémentation efficace du paradigme MapReduce au sein des systèmes HPC. Des industries ont également tenté de proposer des alternatives plus sophistiquées à Hadoop. C'est le cas de Microsoft qui a proposé Dryad dans [13]. Cette alternative propose les mêmes caractéristiques qu'Hadoop avec la possibilité d'ajouter des couches alternatives entre la fonction *Map* et *Reduce*. Cette initiative comme tant d'autres s'est vue abandonné au profit du vaste écosystème Hadoop. Malgré tout, Microsft a sut réagir et proposer une implémentation alternative de Dryad pour Hadoop. Pour ce faire, les ingénieurs se sont reposés sur le nouveau moteur d'exécution d'Hadoop appelé YARN, [14]. Yarn est une brique forte de Hadoop2 en apportant la décomposition de la fonction de *JobTracker* d'Hadoop. Cette démarche permet de mieux répartir les calculs et les exécutions de tout type de modèle de calcul. Le modèle *Map/Reduce* n'est alors plus le seul modèle possible. Comme le présente Romel dans [15], cette nouvelle méthodologie permet de déployer des clusters Hadoop sur des environnements de très grande taille : supérieure à 10.000 nœuds.

L'adoption d'Hadoop est largement prouvée est de nombreux grands noms des industries informatiques proposent maintenant leur propre distribution Hadoop adaptée à des besoins spécifiques. Dans [16], Intel propose une version spécifique d'Hadoop optimisés pour le matériel INTEL (paramètres optimaux pour le SSD, le BIOS, etc.). Microsoft présente également une distribution spécifique pour leur écosystème logiciel, [22]. Malgré tout, de nouveaux acteurs ont réussi à émerger grâce à ce nouvel écosystème, et trois fournisseurs sont principalement impliqués dans ce nouvel environnement. Il s'agit de Ambari (19), Cloudera (18) et HortonWorks (17). Les deux dernières proposent une distribution clé en main avec une assistance technique et de nombreux outils de gestion avancée. De plus, ces deux fournisseurs proposent également un bac à sable basé sur une machine virtuelle pour découvrir cet environnement. Le troisième (Ambari), quant a lui, propose un outil de déploiement d'une infrastructure Hadoop en utilisant un ensemble de machines virtuelles. L'outil fournit également tous les besoins en matière d'outils et d'extensions. Ces différentes stratégies reposent sur l'utilisation de machines virtuelles et tout naturellement les fournisseurs

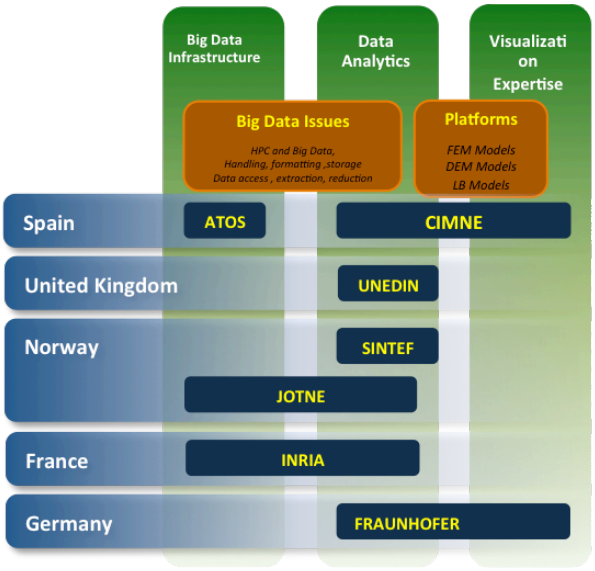

Figure 1. Représentation des différentes compétences des partenaires du projet VELaSSCo

d'hyperviseur se sont mis à proposer leur propre distribution Hadoop, comme par exemple VMware [20].

Dans la section suivante, nous nous intéresserons à présenter le projet VELaSSCo, ainsi que notre outil de déploiement d'une distribution Hadoop en fonction des besoins utilisateurs.

LA PLATEFORME VELASSCO

Dans cette section, nous allons présenter les détails liés au projet VELaSSCo, et par la suite nous présenterons notre outil de déploiement d'écosystème Hadoop.

LE PROJET VELASSCO

Les différentes communautés scientifiques ont une forte tendance au cloisonnement, peu de ces communautés cherchent à interagir avec les communautés voisines.

Des interactions efficaces entre les communautés permet souvent de produire des résultats pertinents en matière de recherche, comme par exemple la combinaison entre informaticien et physicien : les informaticiens vont permettre de produire des moteurs de simulations physique bien plus rapides que ce qu'un physicien pourrait produire. Ainsi, la somme des compétences permet souvent d'atteindre un but plus complexe.

Dans le projet VELaSSCo, l'équipe de recherche est composée de différents intervenants de nombreuses communautés. Notre consortium de plusieurs universités/laboratoires :

- Université d'Edimbourg,
- Le CIMNE (Barcelone),
- SINTEF (Oslo),
- Fraunhofer (Francfort),
- INRIA (Grenoble).

Mais également de deux industriels : ATOS (Espagne) et JOTNE (Norvège).

Ces différents partenaires amènent ainsi leur expertise pour atteindre un but commun : réaliser une plateforme de stockage et de traitement de données massives pour le domaine de l'ingénierie. En Figure 1, nous présentons les différentes compétences apportées par les différents membres de ce consortium.

Ici, trois sous-domaines sont ciblés :

- L'infrastructure BigData
- L'analyse de données
- Et la visualisation d'information.

Pour l'infrastructure BigData, nous avons pour objectif de fournir tous les moyens de stocker et d'appliquer des calculs de manière efficace aux données stockées. Lorsque l'on parle de calcul, il s'agit d'algorithmes qui seront appliqués par la deuxième couche du projet (analyse de données). Cette infrastructure doit être la plus efficace possible et répondre à un certain nombre de besoins : des prérequis matériels, des besoins de stockage, des besoins de complexité de calcul, des besoins de sécurité et des besoin d'extensibilité.

Le matériel disponible par les communautés scientifiques est largement variable. En effet, la plupart disposent d'un centre de calculs, composés de nœuds de calculs haute performance, avec peu de stockage local, toutes les données sont gérées par un partage réseau (par exemple GPFS, [29]). L'accès à ces nœuds est également régit par un gestionnaire de tâches spécifique (par exemple Slurm, Torque, ...). Dans le cas du BigData, l'approche traditionnelle repose sur l'utilisation de nœuds de calcul de bureau pour répondre aux besoins de calcul hautement distribué. Avec cette stratégie, il est possible de constituer une flotte importante de machines. Mais chaque nœud dispose que d'une capacité de calcul très limitée. Le stockage, quant à lui, est réalisé localement. Ces deux architectures sont totalement décolérées et une distribution Hadoop standard ne sera pas en mesure de gérer le plus efficacement les capacités de l'une ou l'autre des infrastructures.

Le deuxième besoin lié à notre infrastructure repose sur la stratégie de stockage. Dans le cadre de notre projet, la solution la plus adaptée semble être de supporter le maximum des solutions de stockage. En effet, un de nos partenaires impose l'utilisation de sa solution de stockage pour offrir une version commerciale de notre plateforme. Ainsi, il est nécessaire d'adopter une stratégie extensible en utilisant toutes les capacités d'Hadoop. Grâce à la couche d'abstraction fournit dans Hadoop, il est possible à notre partenaire de développer un plug-in d'interconnexion entre Hadoop et leur solution de stockage.

La plateforme que nous allons développer nécessite d'être capable de supporter des analyses plus ou moins

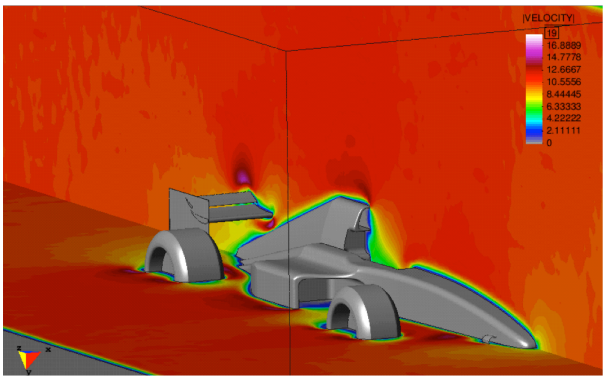

Figure 2. Résultat d'une simulation FEM.

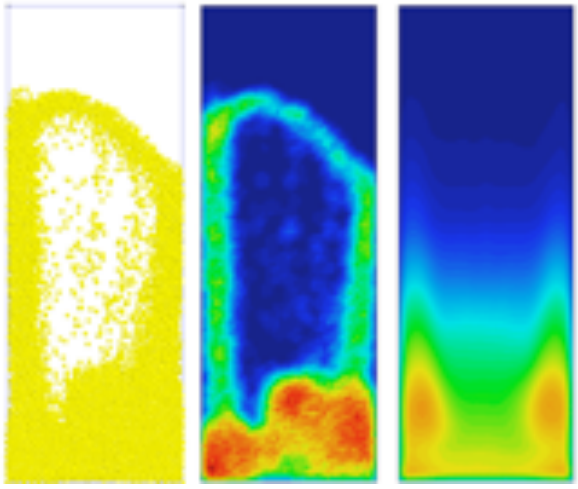

Figure 3. Résultat d'une simulation de fluide.

complexes sur les données. Ainsi, cette plateforme doit être en mesure de supporter des pipelines de calculs plus ou moins complexes. Avec la nouvelle version d'Hadoop, ce problème est résolu, en effet, Yarn amène la capacité d'appliquer n'importe quel type de modèle de calcul sur une plateforme Hadoop [14].

Pour les besoins en sécurité, il est nécessaire d'apporter une solution qui protège l'accès aux données stockées sur la plateforme. Tous les utilisateurs, ne doivent pas pouvoir accéder à toutes les données ; en effet, lorsque l'on traite avec des données de l'ingénierie, il est nécessaire de rendre confidentiel un certain nombre de documents.

Enfin la plateforme doit être extensible pour supporter au mieux les développements qui seront nécessaires. Le développement incrémental de la plateforme nécessite que les différentes briques soient mises en place au cours du temps, et donc que la plateforme supporte leur arrivée progressive. Cette extensibilité permettra à la plateforme de proposer une solution open-source et close-source (basée sur des outils propriétaires).

Maintenant, nous allons nous attacher à présenter les données qui seront utilisées dans le cadre de ce projet.

Ici, nous allons nous intéresser à deux types spécifiques de données issues de deux types de simulations : FEM et DEM.

Une simulation FEM est basée sur une décomposition spéciale d'un environnement. Le moteur de simulation produit des données pour chaque sommet de cette décomposition. Ainsi, il est possible de produire une visualisation volumique d'un espace. Un exemple de rendu possible est présenté en Figure 2. Ici, les couleurs représentent l'impact des turbulences produites par une Formule 1. Dans le cas de simulation FEM, le résultat produit par ces simulateurs est sous forme de tableaux.

Les simulations FEM sont généralement appelées simulation à base de particules. Ainsi, un certain nombre de petits éléments sont utilisés (particules), est interagissent les uns avec les autres (moteur de simulation). Cette stratégie est régulièrement utilisée pour réaliser des simulations de fluide. Le rendu d'une telle simulation est proposé en Figure 3. Comme précédemment, la donnée produite par les simulateurs est composée d'éléments sous la forme d'un tableau. Un extrait de ce résultat est présenté ci-dessous :

```
TIMESTEP PARTICLES

0 868019

ID GROUP RADIUS MASS PX PY PZ VX VY VZ Angular_Velocity_X Angular_Velocity_Y
Angular_Velocity_Z

1 1 5.3705e-05 0.198709 0.0233948 -0.105028 0.0233836 -3.56961e-05 -5.10717e-05 9.75788e-
06 0.00106043 0.000390757 -0.000536836

2 1 5.74099e-05 0.212416 0.0865132 -0.104191 0.0238821 -0.000101408 -5.40777e-05 -
0.000558674 -0.0015792 -0.00176641 0.00125497

…
```

La plateforme que nous devons mettre en place devra ainsi gérer différents types de données :

- Des modèles 3D (objets de CAO, terrain, etc.),
- Des métadonnées (description des modèles, propriétaire, etc.)
- Des pas de temps de simulations.

Comme nous l'avons expliqué précédemment, nous allons pour ces travaux utiliser une distribution Hadoop configurable. Cette distribution s'appuiera sur une succession de couches interagissant les unes avec les autres. Une vue d'ensemble de la plateforme ainsi que les différentes stratégies de communication utilisées est présentée en Figure 4.

Pour comprendre cette figure, il est nécessaire de passer par un exemple d'utilisation.

Un utilisateur lance une simulation sur des nœuds de calcul spécifiques. Ces nœuds de calculs produisent des fichiers correspondant à des pas de simulation. Ces fichiers sont récupérés par la plateforme VELaSSCo au travers d'agents *Flume* dédiés. Ces agents utilisent la couche d'abstraction du système de fichiers pour stocker le résultat des simulations (cette stratégie permet d'utiliser différentes méthodes de stockage des données).

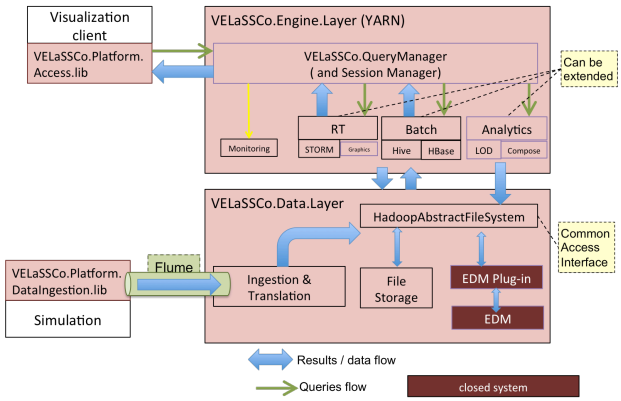

Figure 4. VELaSSCo plateforme.

Une fois cette donnée stockée, l'utilisateur peut y accéder. Pour ce faire, grâce à son outil de visualisation, l'utilisateur émet une requête, qui est transmise à la plateforme VELaSSCo. Cette requête est ensuite interprétée par une couche de gestion (appelé VELaSSCo-query-manager : VQM). Notre *VQM* va ensuite produire la (ou les) requête(s) nécessaire(s) correspondante(s). Ensuite, le composant impliqué (*RT* pour la couche d'extraction temps réel, *Batch* pour l'extraction de donnés en bloc et *analytics* pour le traitement appliqué aux données) dans la sous-requête la traite en extrayant les informations nécessaires dans la couche d'abstraction du système de fichiers. Enfin le résultat est transmis au VQM puis à l'outil de visualisation.

La décomposition introduite par le VQM nécessite d'utiliser les outils adéquats pour extraire certains types d'informations. Dans cet article, nous nous sommes intéressés aux données produites par les simulateurs (sous la forme de table). Dans la section suivante, nous allons présenter notre outil GDF (**G**eneric **D**eployment tool **F**or Hadoop) qui nous a permis d'évaluer les différents composants nécessaires pour extraire de la plateforme Hadoop ces données.

DETAIL DE NOTRE OUTIL : GDF

Pour ce projet, nous avons mis en place un nouvel outil, similaire à myHadoop [21]. MyHadoop est un outil spécialisé dans le déploiement d'une distribution Hadoop sur des systèmes de calcul haute performance. Il fournit un certain nombre de script pour déployer et exécuter un environnement Hadoop sur ce type d'infrastructure. Cet outil propose deux modes de données : avec ou sans persistance. Pour le mode sans persistance, il est nécessaire de transférer à chaque démarrage, l'intégralité des données sur la plateforme Hadoop déployée. Malgré tout, cette stratégie ne fournit pas de performances optimales, en effet, Hadoop ne supporte pas nativement l'utilisation de multi-cœur. De plus, cet outil ne permet pas de configurer d'extensions supplémentaires, il est nécessaire d'installer celles-ci à la main. Enfin, cet outil

est principalement destiné à supporter les systèmes HPC et non tous les types de systèmes informatiques.

Afin de palier ces différents manques, nous avons mis en place GDF (**G**eneric **D**eployment tool **F**or Hadoop). En se basant sur une description utilisateur de l'architecture désirée, GDF est capable de déployer une infrastructure Hadoop préconfigurée sur n'importe quel type de système informatique. Pour ce faire, cet outil utilise une description utilisateur sous la forme d'un fichier XML qui permet de sélectionner différents paramètres. Grâce à ce fichier, l'utilisateur du service peut spécifier la localisation du déploiement de différents services, quel type de stockage souhaite-t-il utiliser, quel type de virtualisation souhaite-t-il mettre en place, etc. Un aperçu de deux déploiements possibles est présenté en Figure 5 et Figure 6.

Dans la première figure (5), nous avons fait le choix de déployer notre distribution Hadoop sur une infrastructure mixte. Ici, notre système est composé de deux nœuds de calculs : *Shark* et *Shok* (deux machines physiques distinctes). Sur la machine *Shark*, l'utilisateur souhaite déployer 3 machines virtuelles en utilisant *Docker* comme outil de virtualisation. Tandis que sur la machine *Shok*, le déploiement de la virtualisation s'effectue à l'aide de *Virtualbox*. Tous les nœuds virtuelles et *Shok* sont utilisés pour le stockage et comme nœud de calcul *Yarn*. Seul *Shark* n'est pas utilisé dans cette optique, ce nœud est utilisé pour gérer la plateforme Hadoop comme nœud maitre, et également pour supporter les différents services utilisés (service *HBase*, service *Hive* service *NFS* et *PIG*). Pour la partie stockage, le XML décrit l'utilisation du système *HDFS*.

Dans le cadre de la figure 6, l'utilisateur fait le choix de ne pas utiliser de virtualisation. Tous les services nécessaires à cette distribution sont exécutés directement sur les nœuds physiques. De plus afin de répartir les charges, les différents services (*HBase*, *Hive*, etc) sont déployés sur les différents nœuds. Dans ce cadre, aucune stratégie de virtualisation n'est utilisée.

Notre outil est en charge de générer tous les fichiers de configuration nécessaires au déploiement d'une distribution Hadoop sur l'infrastructure décrite. De plus, cet outil produit également les outils (scripts de lancements) nécessaires à la compatibilité de cette plateforme déployée avec un système de gestion de tâches HPC. A l'heure actuelle, seul *Slurmm* est supporté. Ainsi, la plateforme Hadoop déployée peut alors être exécutée sur n'importe quel type d'infrastructure. La connexion entre les différentes machines est quant à elle assurée par la mise en place de réseaux virtuels. De plus afin de permettre d'utiliser au mieux les ressources disponibles, notre solution supporte également le déploiement de machines virtuelles en utilisant pour le moment deux outils : *VirtualBox* [24] et *Dockers* [25]. Dans le premier cas, il s'agit d'un hyperviseur de machines virtuelles, qui nécessite le déploiement et l'installation d'un véritable système d'exploitation. Dans le deuxième cas, il s'agit

```
<?xml version="1.0" encoding="UTF-8"?>
<VELaSSCo>
    <!-- Loc ressrouces Hadoop, Hive, ... -->
    <path>
        <install>/local_home/lange/distrib</install>
        <tools>/local_home/lange/tmp/tools</tools>
    </path>
    <real>
        <node>

            <host>shark</host>
            <hadoop>
                <master></master>
                <hbase></hbase>
                <hive></hive>
                <pig></pig>
                <NFS></NFS>
            </hadoop>
            <virtual>
                <dockers>
                    <host>velassco01</host>
                    <hadoop>
                        <FS>
                            <kind>HDFS</kind>
                            <path>/tmp</path>
                        </FS>
                    </hadoop>
                </dockers>

                <dockers>
                    <host>velassco02</host>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </dockers>
                <dockers>
                    <host>velassco03</host>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </dockers>
            </virtual>
        </node>
        <node>
            <host>shok</host>
            <hadoop>
                <FS>
                </FS>
            </hadoop>
            <virtual>
                <vbox>
                    <host>velassco11</host>
                    <ip>192.168.64.102</ip>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </vbox>
                <vbox>
                    <host>velassco12</host>
                    <ip>192.168.64.103</ip>
                    <hadoop>
                        <FS>
                        </FS>
                    </hadoop>
                </vbox>
            </virtual>

        </node>
    </real>
</VELaSSCo>
```

Figure 5 Exemple de déploiement complexe (plusieurs nœuds physique et plusieurs nœuds virtuels)

```
<?xml version="1.0" encoding="UTF-8"?>
<VELaSSCo>
    <!-- Loc ressrouces Hadoop, Hive, ... -->
    <path>
        <install>/local_home/lange/distrib</install>
        <tools>/local_home/lange/tmp/tools</tools>
    </path>
    <real>
        <node>

            <host>shark</host>
            <hadoop>
                <master></master>
                <pig></pig>
                <NFS></NFS>
                <FS>
                    <kind>local</kind>
                    <path>/mnt/tmp</path>
                </FS>
            </hadoop>
        </node>
        <node>
            <host>shok</host>
            <hadoop>
                <FS></FS>
                <hbase></hbase>
            </hadoop>
        </node>

    <node>
        <host>sheep</host>
        <hadoop>
            <FS></FS>
            <hive></hive>
        </hadoop>
    </node>
</real>

</VELaSSCo>
```

Figure 6. Exemple de déploiement sur des machines hosts.

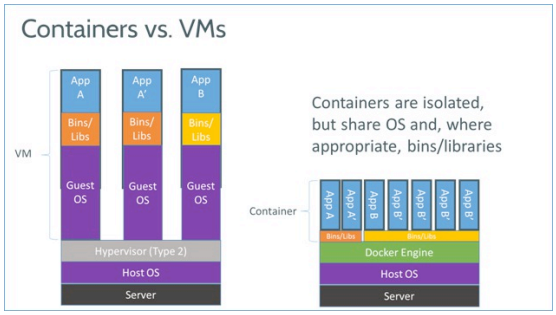

Figure 6. Comparaison entre machines virtuelles et conteneurs (extrait de 30).

simplement de conteneurs linux, qui permettent le déploiement de machines virtuelles avec une surconsommation de calcul réduite, cf. Figure 6. Cette différence de performances a également été présentée dans [28].

EVALUATION DE NOTRE PLATEFORME

Afin de valider notre approche, nous avons mis en place une série de mesure. Ces mesures sont basées sur des requêtes spécifiques nécessaires pour notre projet. Nous ne nous sommes pas basés sur des outils de mesure existantes tel que HiBench (26) ou l'utilisation des Hadoop Blast (27). Ces outils imposent trop de contraintes logistiques (version et configuration

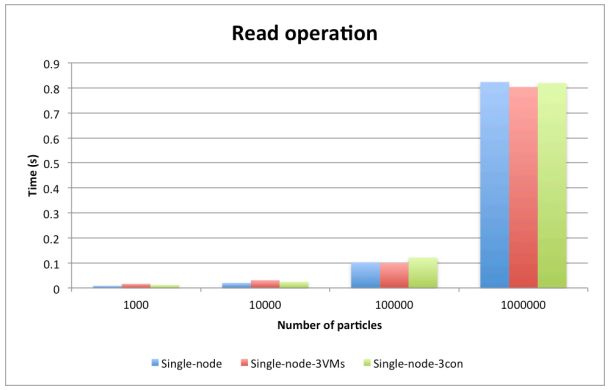

Figure 8. Operations de lecture partielle pour le pont NFS.

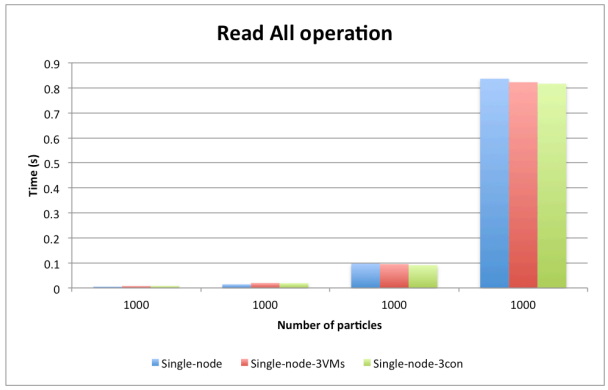

Figure 9. Operations de lecture totale pour le pont NFS.

principalement). Ainsi, nous avons mis en place notre propre solution de mesure basée sur deux types de requêtes : extraction complète d'un pas de temps et sélection d'un sous-ensemble de données de ce même pas de temps. Nous nous intéressons seulement à évaluer les performances des extensions Hadoop pour la lecture, car la phase d'écriture sera gérée par les agents *Flume*. Pour notre évaluation, nous avons utilisé un jeu de données composé de 10.000, 100.000 et 1.000.000 de particules, avec pour chaque particule : leur identifiant, leurs coordonnées en *X, Y, Z*, leur accélération (*X, Y, Z*) et leur vitesse (*X, Y, Z*). Pour l'opération de lecture partielle, les données seront extraient en fonction de l'identifiant des particules.

Pour cette évaluation, nous avons utilisé un seul nœud de calcul composé d'un processeur Intel Core i7-2620M. Ce processeur possède 2 cœurs physiques hyper-threadés. Cette machine est également équipée de 8 Go de mémoire vive. Dans cette expérimentation, nous avons évalué 3 stratégies d'accès aux données :

- En utilisant le pont *NFS* fourni pas Hadoop
- En utilisant le server *Thirft* (23) de *HBase*
- En utilisant le server *Thrift* de *Hive*.

Nous avons également évalué la pertinence d'utiliser notre distribution sur 3 configurations : soit en utilisant

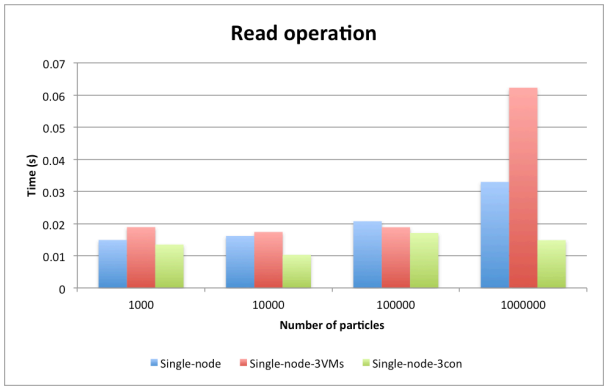

Figure 10. Operations de lecture partielle pour HBase.

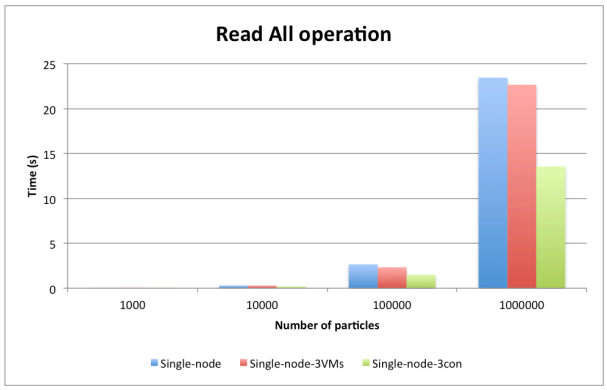

Figure 11. Operations de lecture totale pour HBase.

un seul nœud Hadoop, soit en utilisant 3 machines virtuelle en utilisant *Virtualbox,* soit en utilisant 3 machines *Dockers* comme machine virtuelle.

Pour la partie client/visualisation, nous avons développé un outil générique de visualisation de particules en C++, spécialement conçu pour supporter les trois moyens d'accès aux données Hadoop.

Les figures 8 et 9 présentent les temps de réponse des trois configurations matérielles en utilisant le pont *NFS*. Un point de montage (vers le stockage *HDFS*) est réalisé sur la machine de visualisation et l'accès aux données est réalisé par lecture de fichiers. Quelle que soit l'architecture, aucune méthode en lecture partielle ou totale ne se distingue. En effet, un seul nœud est utilisé pour transférer les informations vers le client. Avec cette méthodologie (point de montage *NFS*) il est nécessaire de parcourir l'intégralité du fichier de données (même pour le filtrage) car les particules ne sont pas ordonnées. Ceci explique la non différence de temps d'accès entre l'opération de lecture globale et l'opération de lecture partielle.

Pour les figures 10 et 11, nous présentons les résultats obtenus pour l'accès à des données stockées dans une table *HBase*. Ici, la solution basée sur l'utilisation de 3 conteneurs est la plus efficace, le temps d'extraction et de transfert est le plus court que ce soit pour les opérations de lecture globale ou partielle (dans le cadre d'un grand

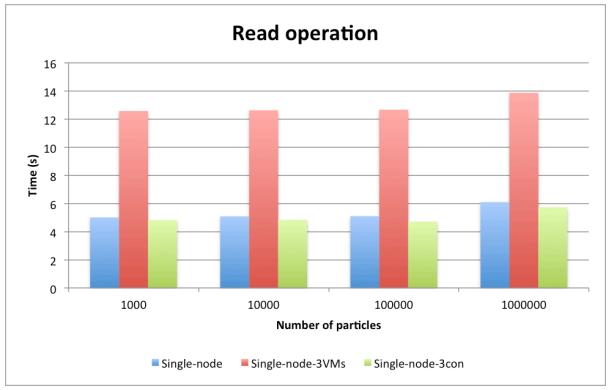

Figure 12. Operations de lecture partielle pour le plugin Hive.

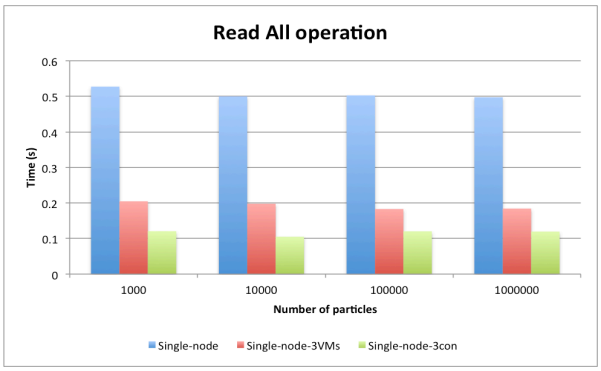

Figure 13. Operations de lecture totale pour le plugin Hive.

nombre de particules). La solution basée sur des machines virtuelles souffre d'un fort ralentissement lors de la lecture partielle en comparaison avec le déploiement effectué directement sur la machine hôte. Ainsi, afin d'obtenir le meilleur temps de réponse avec *HBase*, il est nécessaire d'utiliser des conteneurs.

Enfin, nous avons étudié la faisabilité d'utilisation du plug-in *Hive*. Ces résultats sont présentés dans les figures 12 et 13. En utilisant le plugin *Hive*, les temps d'accès sont similaires quelle que soit la stratégie utilisée. Une des solutions se démarque le plus, il s'agit de l'utilisation des 3 conteneurs. L'utilisation de machines virtuelles est plus efficace que le déploiement sur la machine hôte pour la lecture totale du jeu de données. Dans le cas de la sélection, les performances des machines virtuelles basées sur *Virtualbox* s'effondrent.

De ces trois évaluations nous pouvons extraire un certain nombre de remarques vis à vis des besoins du projet VELaSSCo. La stratégie la plus efficace pour accéder aux données semble basée sur une approche hybride entre le plugin *HBase* (pour extraire des données partielles) et *Hive* (pour extraire l'ensemble des données). L'utilisation du pont *NFS* ne semble pas pertinente pour notre cas d'utilisation. Un des avantages de *HBase* est qu'il peut être combiné à *Hive* en accédant

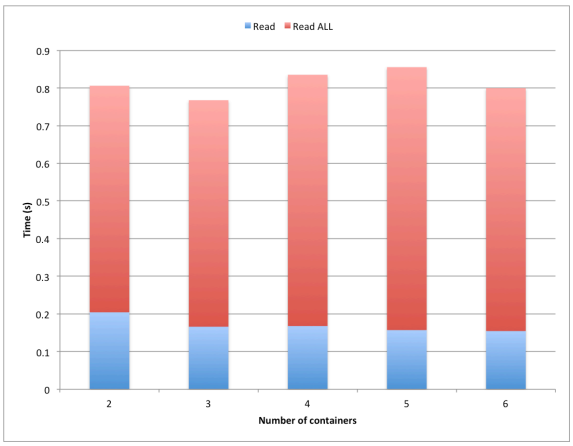

Figure 7.Evaluation d'une stratégie mixte HBase/Hive en fonction du nombre de conteneurs utilisés.

aux données stockées dans *HBase* au travers de l'API *Hive* (requête *HiveQL* appliquée aux données *HBase*).

De plus, l'architecture la plus efficace pour effectuer nos traitements semble être l'utilisation de conteneurs, puis l'utilisation des machines hôtes directement et enfin de machines virtuelles avec hyperviseur. Ce troisième type de méthode amène un surplus important en terme de temps de calcul mais pas seulement. En effet, si on compare les temps de déploiement des trois approches, *Virtualbox* a nécessité 188.2 secondes pour obtenir un système opérationnel, les conteneurs *Dockers* sont opérationnels en 8.8 secondes tandis que déployées une distribution Hadoop sur une machine hôte nécessite seulement 1.7 secondes.

Une fois la meilleure stratégie évaluée, nous avons proposé une évaluation du nombre de conteneurs idéale pour notre solution mixte *Hive*/*HBase*. Cette évaluation est présentée en figure 14. Ainsi, la méthode la plus efficace est d'utiliser 3 conteneurs.

CONCLUSIONS ET PERSPECTIVES

Dans cet article, nous avons présenté une vue d'ensemble du projet VELaSSCo ainsi que notre nouvel outil de déploiement d'une distribution Hadoop spécifiquement développé pour ce projet.

Ce projet européen cherche à développer une nouvelle solution de stockage de données spécialement conçue pour stocker des données issues de calculs scientifiques dans le cadre de l'ingénierie. Cette plateforme sera spécialement destinée à stocker deux types de données issues de simulations FEM et DEM. Ces deux jeux de données sont produits par des outils de simulation spécifiques. Ces outils de simulation sont exécutés sur des nœuds de calculs HPC. Malheureusement ce type de nœud de calculs n'est pas spécialement défini pour supporter le stockage de données massives. Traditionnellement le stockage pour le BigData est principalement réalisé par des systèmes de stockage dédiés. Dans le cadre de ce projet, nous cherchons à mettre en place une solution de stockage éligible à n'importe quel type d'infrastructure (que ce soit un centre de calcul HPC ou un centre de stockage).

Afin de valider différents composants nécessaires pour stocker les données du projet, nous avons évalué différents composants de la distribution Hadoop. Pour ce faire, nous avons étudié le comportement de trois plugins : *NFS*, *Hive* et *HBase*. Nous avons également évalué la faisabilité d'accélérer les performances de notre distribution en utilisant une stratégie de virtualisation. Les résultats de notre étude ont montré que pour nos données de simulation, il est nécessaire d'utiliser une stratégie hybride entre *Hive* et *HBase* pour obtenir le meilleur temps de réponse pour des requêtes. De plus, pour augmenter les capacités de calcul de notre solution, l'utilisation de conteneurs semble la plus efficace.

De futurs développements sont encore à prévoir, notamment afin de pouvoir distribuer notre outil de déploiement de distribution Hadoop. Il est nécessaire pour cela de passer par une phase de tests des différents scripts produits avec plusieurs configurations matérielles.

De plus, afin de répondre aux attentes de notre projet, il sera également nécessaire d'évaluer certains éléments : comme les composants temps réel d'Hadoop (avec *Storm*), l'extensibilité en proposant une nouvelle méthode de stockage de données basée sur la solution EDM de JOTNE. Mais également une extensibilité des composants d'analyse en implémentant les différentes méthodes d'analyse relative à des données d'ingénierie (extraction de skin, de composantes, spline etc.)

Enfin, il est nécessaire de faire évoluer les mentalités des administrateurs systèmes. En effet, la plupart des systèmes de calcul haute performance ne sont pas compatibles avec l'utilisation de méthodes de virtualisation. Il est ainsi nécessaire de mettre en place ce type de stratégies afin d'augmenter les capacités de calcul des systèmes distribués. Malgré tout, cette méthode ne doit pas être réalisée de n'importe quelle manière : la sur-utilisation des machines virtuelles avec un hyperviseur lourd semble amener un coût relativement important comparé aux stratégies basées conteneurs.

REMERCIEMENTS

Nous souhaitons remercier l'Union Européenne pour son support financier au travers de l'instance FP7 et de notre projet numéro : 619439, FP7-ICT-2013-11.

Nous souhaitons également remercier les différents membres du consortium : CIMNE, UEDIN, SINTEF, Fraunhofer, JOTNE and ATOS

BIBLIOGRAPHIE

1. S. Tansley & K. M. Tolle, The fourth paradigm: data-intensive scientific discovery, 2009.
2. B. Lange & P. Fortin, "Parallel dual tree traversal on multi-core and many-core architectures for astrophysical n-body simulations," Euro-Par 2014, 2014.

3. W. Fan and A. Bifet. Mining big data: current status, and forecast to the future. ACM SIGKDD Explorations Newsletter, 14(2):1 to 5, 2013.
4. D. Laney. 3d data management: Controlling data volume, velocity and variety. META Group Research Note, 6, 2001.
5. J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Commun. ACM, 51(1):107 to 113, Jan. 2008.
6. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008
7. Ghemawat, S., Gobioff, H., & Leung, S. T. (2003, October). The Google file system. In ACM SIGOPS Operating Systems Review (Vol. 37, No. 5, pp. 29-43). ACM.
8. K. Saroj. Idc report to hadoop leads the big data analytics tool for enterprises - http://cloudtimes.org/2013/11/06/idc-report-hadoop-leads-the-big-data-analytics-tool-for-enterprises/.
9. D. Borthakur, The hadoop distributed file system: Architecture and design. Hadoop Project Website, 11, 21, 2007.
10. C. Lam, Hadoop in action. Manning Publications Co., 2010.
11. M. N. Vora, Hadoop-HBase for large-scale data. In Computer Science and Network Technology (ICCSNT), 2011 International Conference on (Vol. 1, pp. 601-605). IEEE, 2011.
12. R. C. Taylor, An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC bioinformatics, 11(Suppl 12), S1, 2010.
13. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operating Systems Review, 41(3):59 to 72, 2007.
14. V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. Oâ€™Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC'13, New York, NY, USA, 2013. ACM.
15. G. Rommel, Hadoop 1.x vs Hadoop 2.
16. Intel Distribution for Apache Hadoop* Software: Optimization and Tuning Guide
17. http://hortonworks.com
18. http://www.cloudera.com
19. http://ambari.apache.org
20. Virtualizing Apache Hadoop, VMware, 2012
21. S. Krishnan, M. Tatineni, and C. Baru. myhadoop-hadoop-on-demand on traditional hpc resources. San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego, 2011.
22. http://azure.microsoft.com/en-us/services/hdinsight/
23. http://thrift.apache.org
24. https://www.virtualbox.org
25. https://www.docker.com
26. https://github.com/intel-hadoop/Hibench
27. https://portal.futuregrid.org/manual/hadoop-blast
28. W. Felter, A. Ferreira, R. Rajamony, J. Rubio, An Updated Performance Comparison of Virtual Machines and Linux Containers, IBM Research Division, 2014
29. http://www-03.ibm.com/systems/platformcomputing/products/gpfs/
30. http://koin.github.io/what-is-docker/